CLAIMS

What is claimed is:

1. A method for storing unstructured XML data into a relational database, comprising:

assigning a document identifier to an XML document;

parsing the XML document to identify a node;

for the identified node in the XML document:

storing path information for the node;

storing hierarchical information for the node; and

storing node data for the node.

2. The method of claim 1 in which the hierarchical information comprises a hierarchical level within the XML document.

3. The method of claim 1 in which the node data comprises a start position, end position, node type, or node value.

4. The method of claim 1 in which the document identifier is a unique identifier for each different XML document.

5. The method of claim 1 in which the path information comprises a full path for the node.

6. The method of claim 1 in which the path information comprises a path identifier.

7. The method of claim 6 in which the path identifier corresponds to a key to a path entry containing a full path for the node.

8. The method of claim 7 in which the path entry resides in a first table structure and the path information, hierarchical information, and node data reside in a second table structure.

9. The method of claim 7 in which the path entry comprises node name corresponding to a name of a terminal node.

10. The method of claim 1 further comprising:

maintaining one or more indexes.

11. The method of claim 10 in which the one or more indexes comprise an index on a path identifier, an index on the document identifier and a start position, or an index on the document identifier, start position, and node level.

12. The method of claim 10 in which the path identifier corresponds to a key to a path entry containing a full path for the node, the path entry resides in a separate table, and the one or more indexes comprise an index on path identifiers or a unique index on reverse paths.

13. A computer-implemented structure for storing XML data in a relational database, the computer implemented structure comprising a first table structure, the first table structure comprising:

a document identifier corresponding to an XML document;

path information for a node within the XML document;

hierarchical information for the node; and

node data for the node.

14. The computer-implemented structure of claim 13 in which the hierarchical information comprises a hierarchical level within the XML document.

15. The computer-implemented structure of claim 13 in which the node data comprises separate columns for a start position, end position, node type, or node value.

16. The computer-implemented structure of claim 13 in which the document identifier is a unique identifier for each different XML document.

17. The computer-implemented structure of claim 13 in which the path information comprises a full path for the node.

18. The computer-implemented structure of claim 13 in which the path information comprises a path identifier.

19. The computer-implemented structure of claim 18 in which the path identifier corresponds to a key to a path entry in a second table structure.

20. The computer-implemented structure of claim 19 in which the path entry comprises a full path for the node.

21. The computer-implemented structure of claim 18 in which the path entry comprises a node name corresponding to a name of a terminal node.

22. A method to access a computer-implemented structure for storing XML data in a relational database, the computer implemented structure comprising a first table structure, the first table structure comprising a document identifier corresponding to an XML document, path information for a node within the XML document, hierarchical information for the node, and node data for the node, the method comprising:

generating a SQL query against the computer-implemented structure; and

producing a result set based upon executing the SQL query.

23. The method of claim 22 in which the SQL query reconstructs the XML document.

24. The method of claim 23 in which the SQL query provides the same result as the following:

```
select i.nodename, p.startpos, p.endpos, p.nodetype, p.nodeval
from path_table p, path_index_table i
where p.docid = :1 and p.pid = i.pid
```

```
order by p.startpos
```

where path_table comprises a first column for the start position of the node (startpos), a second column for the end position of the node (endpos), a node type column (nodetype), a node value column (nodeval), a path identifier column (pid), and a document identifier column (docid), and a path_index_table comprises a path identifier column (pid), a path column (path), and a nodename column (nodename).

25. The method of claim 22 in which the SQL query identifier a fragment within the XML document.

26. The method of claim 25 in which the SQL query provides the same result as the following:

```
select i.nodename, p.startpos, p.endpos, p.nodetype, p.nodeval
from path_table p, path_index_table i,
 (select docid, startpos, endpos from path_table
  where rowid = :1) p2
where p.docid = p2.docid and p.startpos >= p2.startpos
and p.endpos <= p2.endpos and p.pid = i.pid
order by p.startpos
```

where path_table comprises a first column for the start position of the node (startpos), a second column for the end position of the node (endpos), a node type column (nodetype), a node value column (nodeval), a path identifier column (pid), and a document identifier

column (docid), and a path_index_table comprises a path identifier column (pid), a path column (path), and a nodename column (nodename).

27. The method of claim 22 in which the SQL query corresponds to an XPath expression.

28. The method of claim 27 in which the XPath expression is translated to the SQL query by:

breaking the XPath expression into multiple components;

creating a new SQL query corresponding to each of the multiple components; and

joining the new SQL query corresponding a component to its previous component.

29. The method of claim 28 in which the XPath expression is broken into multiple components by considering each continuous segment of simple XPath, wherein each occurrence of a predicate within the XPath causes creation of a new component.

30. The method of claim 29 wherein a set of node names separated by "/" corresponds to a single XPath component.

31. The method of claim 28 in which the new SQL query comprises a join of a path_index_table and a path_table.

32. The method of claim 28 in which the new SQL query comprises one or more conditions.

33. The method of claim 32 in which the one or more conditions comprises a condition for the path being chosen, a condition for the node type, or a condition for the node value.

34. The method of claim 28 in which the act of joining the new SQL query corresponding the component to its previous component uses a join condition comprising a join on a document identifier or a join on a hierarchy relationship.

35. A method for managing an unstructured document in a relational database system, comprising:

storing the unstructured document in a storage structure in the relational database system, the storage structure corresponding to a universal schema;

determining whether to create an index upon the storage structure, wherein one or more indexes are maintained if desired; and

accessing the unstructured documents by accessing the storage structure.

36. The method of claim 35 in which the unstructured document comprises an XML document.

37. The method of claim 36 in which the storage structure comprises:

a document identifier corresponding to an XML document;

path information for a node within the XML document;

hierarchical information for the node; and

node data for the node.

38. The method of claim 37 in which the one or more indexes comprise an index on a path identifier, an index on the document identifier and a start position, or an index on the document identifier, start position, and node level.

39. The method of claim 36 further comprising a second structure for storing path data, the second structure comprising:

a path identifier;

a full path for the node; and

a node name corresponding to a name of a terminal node.

40. The method of claim 39 in which the one or more indexes comprise an index on path identifiers or a unique index on reverse paths.

41. The method of claim 35 in which the unstructured documents are accessed by accessing the storage structure using a SQL query.

42. The method of claim 41 in which the SQL query reconstructs the XML document.

43. The method of claim 41 in which the SQL query identifier a fragment within the unstructured documents .

44. The method of claim 41 in which an XPath expression is translated to the SQL query by:

breaking the XPath expression into multiple components;

creating a new SQL query corresponding to each of the multiple components; and

joining the new SQL query corresponding a component to its previous component.

45. A computer program product comprising a computer usable medium having executable code to execute a process for storing unstructured XML data into a relational database, the process comprising:

assigning a document identifier to an XML document;

parsing the XML document to identify a node;

for the identified node in the XML document:

storing path information for the node;

storing hierarchical information for the node; and

storing node data for the node.

46. A system for storing unstructured XML data into a relational database, comprising:

means for assigning a document identifier to an XML document;

means for parsing the XML document to identify a node;

for the identified node in the XML document:

means for storing path information for the node;

means for storing hierarchical information for the node; and

means for storing node data for the node.

47. A computer program product comprising a computer usable medium having executable code to execute a process to access a computer-implemented structure for storing

XML data in a relational database, the computer implemented structure comprising a first table structure, the first table structure comprising a document identifier corresponding to an XML document, path information for a node within the XML document, hierarchical information for the node, and node data for the node, the process comprising:

generating a SQL query against the computer-implemented structure; and

producing a result set based upon executing the SQL query.

48. A system to access a computer-implemented structure for storing XML data in a relational database, the computer implemented structure comprising a first table structure, the first table structure comprising a document identifier corresponding to an XML document, path information for a node within the XML document, hierarchical information for the node, and node data for the node, the method comprising:

means for generating a SQL query against the computer-implemented structure; and

means for producing a result set based upon executing the SQL query.

49. A computer program product comprising a computer usable medium having executable code to execute a process for managing an unstructured document in a relational database system, the process comprising:

storing the unstructured document in a storage structure in the relational database system, the storage structure corresponding to a universal schema;

determining whether to create an index upon the storage structure, wherein one or more indexes are maintained if desired; and

accessing the unstructured documents by accessing the storage structure.

50. A system for managing an unstructured document in a relational database system, comprising:

means for storing the unstructured document in a storage structure in the relational database system, the storage structure corresponding to a universal schema;

means for determining whether to create an index upon the storage structure, wherein one or more indexes are maintained if desired; and

means for accessing the unstructured documents by accessing the storage structure.